CSC323 2010S *Software Design*

Class 06: Gathering Requirements

Held: Thursday, February 11, 2010

Summary: We begin to consider ways in which programmers and clients discuss the components of a system, particularly through requirements and use cases.

Related Pages:

* EBoard.

Notes:

* Readings for Tuesday: HFOOAD Chapter 4, Beautiful Code Chapter 30.
* Due time for readings: 9 a.m.
* Next small assignment: Use Cases.
* Class will end at 2:45 today.
* Many of you missed a good convo today.
* EC for Today's CS Extra.
* EC for Tomorrow's CS Table.
* EC for next week's Media Symposium.

Overview:

* About requirements.
* About use cases.
* The dog door example.
* Activity: Use cases for grinnell.edu.

Requirements: The Basics

The big question: *What is the software supposed to do?*

* We're exploring tools and methodlogies that help us ensure that the client and the programmer have a common understanding of the answer to this question.

Most frequently, we call the agreed-upon characteristics of the software the *requirements*.

* Some requirements are *functional*: the software must be able to do x or y or z.
* Some requirements are *contextual*: it must run on this platform or use this language or ....
* Some requirements are *external*: it must be done for this cost, or by this date, or ....

Side notes:

- These are my own classifications.
- Engineering is often considered the art of balancing requirements
- We will focus primarily on functional requirements.
- With some experience, you can use functional requirements (or the tools we use to understand functional requirements) to better estimate time and cost.

Use Cases: A Technique for Exploring Requirements

- Clients tend to think in more vague terms ("I want something that lets my dog out"; "I want a good Web site".)
- Some clients tend to have requirements that may not necessarily make sense for the project at hand ("I want a programmable one-button remote.")
- Hence, we need tools that help formalize these client goals and tie them to the particular requirements we have for the software.
- A *use case* is a story that tells you about how the software acheives one of the client's goals.
- Use cases focus more on *what* the software does than *how* it acheives the goal.
- Ideally,
  - Every requirement handles at least one step of the story. (That is, the requirement is *necessary*.)
  - Every step in the story can be handled by one of the requirements. (That is, the set of requirements is *sufficient*.)

Detour: Dog Doors and Other Issues from HFOOAD

- There were some concerns about the design of the dog door code. We'll consider those a bit.
- We'll also go over any other questions you have.

Use Cases: An Exercise

- Let's consider use cases from the perspective of the Grinnell Web site.
- An abstract goal: "It should be easy to find people."
  - One concrete version of the goal: "If I know someone's name, it should be possible to find their contact information in under N steps."
  - Another concrete version of the goal: "If I know some approximate information about someone's name, and I've met them, it should be possible to find their contact information (and to verify that the name matches the face) in under N steps."
  - Another concrete version of the goal: "If I want someone with expertise in a particular field (say, functional programming), it should be possible to find a name and contact information in under N steps."
- A more concrete goal: "Faculty from any department should be able to write prospective news items."
- Another concrete goal: "There should be a process by which people submit questions and we ensure that those questions get answered."
- Write use cases for a few of these. You can pretend the Web site has features that it currently lacks.

- Identify what requirements these use cases suggest.
- We'll debrief on your use cases after some time has passed.